CS-GY 6643 Computer Vision

Image Semantic Segmentation and Application based on Deep Learning and Computer Vision Technologies

[redacted], Atsushi Shimizu
[redacted], as15106
[redacted], as15106@nyu.edu

Contents

1 Introduction/Motivation

With the development of Artificial Intelligence, people are getting more excited about giving computers the ability to actually see the world just as human beings. We as humans, can easily distinguish edges, objects, colors from digital images. During the journey in this course, we learned algorithms to manipulate images, how the computer/camera sees the object and basic deep learning algorithms. The motivation of this project is that nowadays, people tend to find a tool to manipulating their images/pictures or even videos based on the background and foreground. They can change the background to pretend to be the place; they can blur the background to protect their privacy when they post an image or conduct a zoom meeting. In order to do it, segment the image or detect the object from the video becomes a hot trend. Based on this trend, we would like to explore our own way to segment images.

In this project, we will create our own semantic segmentation model based on deep learning methods that we learnt from the course. Furthermore, we will explore the ways to manipulate the image after acquiring the segmentation. In this report, we will describe our approach for the model, the dataset that we used, the results we get by training the model and the difficulties and challenges that we faced during the project. Finally, we will discuss the strengths and weaknesses of our model and the future work assuming we have more time and knowledge.

2 Approach

Our approach for this project consists basically of two parts; a semantic segmentation task to classify each pixel in an image, and blurring the background of the image using the prediction of the semantic segmentation.

The semantic segmentation is an image classification problem where the class label is assigned to each pixel. We developed our convolutional neural network based on the idea introduced in "U-Net: Convolutional Networks for Biomedical Image Segmentation". There has been many network architectures proposed for semantic segmentation, but we chose this paper's idea because, to give a prediction for one single pixel, it uses both the local information from the neighboring pixels and the context of the entire image. Figure 1 shows the basic idea of the network. The input image is first going through the convolution layers. The output of this part can be thought as the local information. Then, we used a maxpooling layer to reduce the size of image by half. By iterating this convolution and maxpooling again and again, we extract the context of the entire image just as we normally do in the plain image classification. This extracted context is fed into transposed convolution layer (or also called deconvolution), which does the inverse operation of what convolution layer does. Therefore, every time the data goes through this layer, the size

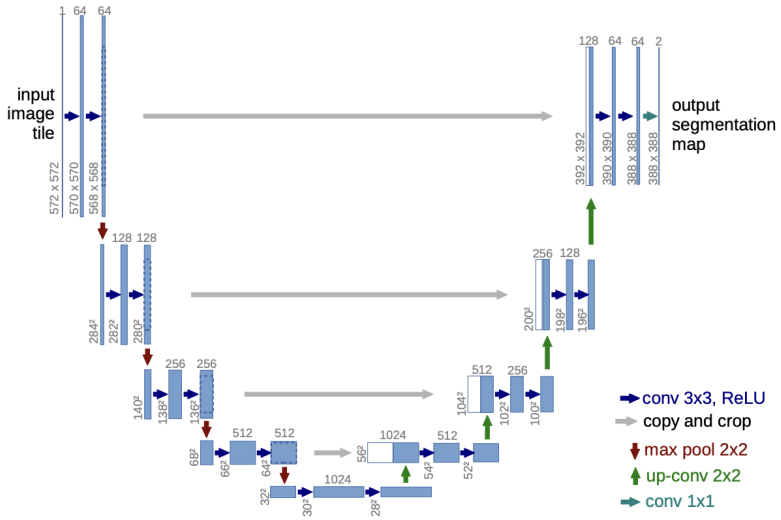


Figure 1: The network architecture from the paper[1]

of the image gets doubled. The trick here is that before we feed this data to another transposed convolutional layer, we put the local information that we obtain in the down-sampling process together. By doing so, we can utilize both the context and the local information for prediction. Although what we actually constructed is different from the network in figure 1, the basic concept is exactly the same as what are presented in this paper.

Using the output of semantic segmentation model above, we blurred the background of the image just like what we often do in the Zoom meeting. To make the background blurred, we used the Gaussian filter. As the Gaussian filter is separable, our implementation is first applying the filter to the vertical direction followed by applying the filter to the horizontal direction in order to speed up the computation.

To improve the prediction accuracy of the semantic segmentation model, we tried a variety of approaches including:

- Hyperparameter tuning:
  The number of filters and the size of kernel in each convolution layer/ the number of convolution layers/ different activation functions/ the use of regularization such as l2 regularization, batch normalization, and dropout/ changing the learning rate and optimizers

- Data augmentation:
  We used data augmentation to increase the size of the training set.

- Histogram equalization:
  We tried histogram equalization as preprocessing. As the images in this project are the RBG color images, we first convert the image to YPbPr scale, apply the histogram equalization only for the Y coordinate, which corresponds to luminance, and convert back to the RGB scale. We refer to the Wikipedia page for YPbPr to implement this.

3 Dataset

The dataset that we used in this project is The Oxford-IIIT Pet Dataset. This dataset contains 37 categories of pets including cats and dogs with roughly 200 images for each class. All the images have associated ground truth annotation of breed, head position, and pixel segmentation. In this project, we will only use the the ground truth and the provided pixel segmentation.

The dataset provides us 3669 images for testing and 3680 images for training.

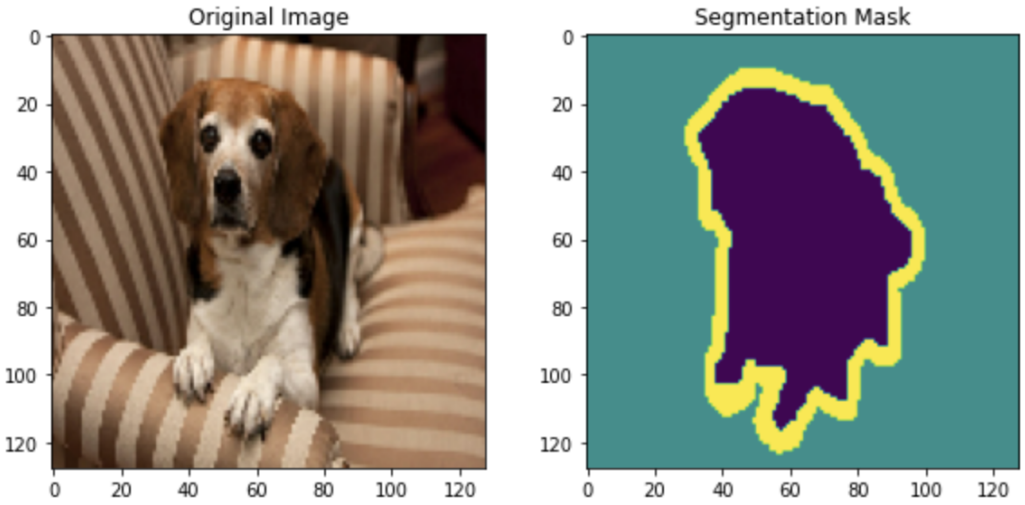


Figure 2: Example Image from Dataset

Figure 2 shows one example image from the dataset. The left side is the ground truth image of a cat with black background. The right side image is the True mask that is provided in the dataset. it separates the image with 3 classed, label 1 for foreground, 2 for background and 3 for not classified. Usually, label 3 is the boundary of the image. Since we are using TensorFlow Keras, we can download the dataset directly using built in functions.

3.1 Data Pre-processing

In order to make the training and testing easier, we need to pre-process the data. Firstly, we will resize the image into shape (128, 128, 3) in order to feed it into our deep network. Then, we need to normalize the image to a range [0, 1] to prevent one variable from being overly influential. For the mask labels, we minus 1 from the labels to make them 0, 1, and 2.

3.2 Data Augmentation

Data augmentation is a decent way to improve the robustness of images when we have a few training data available[1]. In order to improve the robustness, we choose to use TensorFlow's built in random flip function to flip the images with the labels randomly.

4 Results

We tried different network settings to attain a better performance. The rough outline of the best model we got is as follows:

Layers	Output Shape
Input layer	(None, 128, 128, 3)
2 Conv + 1 Maxpool	(None, 64, 64, 32)
2 Conv + 1 Maxpool	(None, 32, 32, 64)
2 Conv + 1 Maxpool	(None, 16, 16, 128)
2 Conv + 1 Maxpool	(None, 8, 8, 256)
2 Conv + 1 Maxpool	(None, 4, 4, 512)
ConvTranspose + Concatenate + 2 Conv	(None, 8, 8, 256)
ConvTranspose + Concatenate + 2 Conv	(None, 16, 16, 128)
ConvTranspose + Concatenate + 2 Conv	(None, 32, 32, 64)
ConvTranspose + Concatenate + 2 Conv	(None, 64, 64, 32)
ConvTranspose + Conv	(None, 128, 128, 3)

Detailed information is in the appendix section and the attached coding. Though it is not shown above, we used the batch normalization to prevent overfitting and used LeakyReLu for activation function. Other major parameter settings are as follows:

Optimizer	Adam with default parameters
Initial learning rate	0.01
Batch size	64
Train epochs	40

Figure 3 shows the loss and accuracy curve during the training. The loss gradually decreases and accuracy increases on the training data, while those on the validation data gets stuck

after about 30 epochs. The accuracy on the training set is 0.9109, while the accuracy on the validation set reaches 0.8788. Unlike the normal image classification problem, it may be

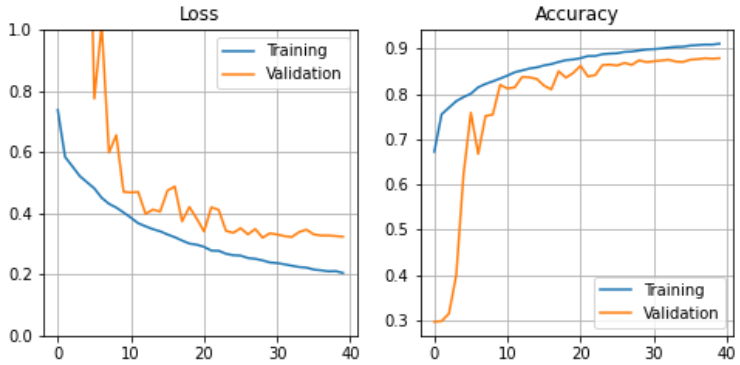


Figure 3: Loss and accuracy curve on the training

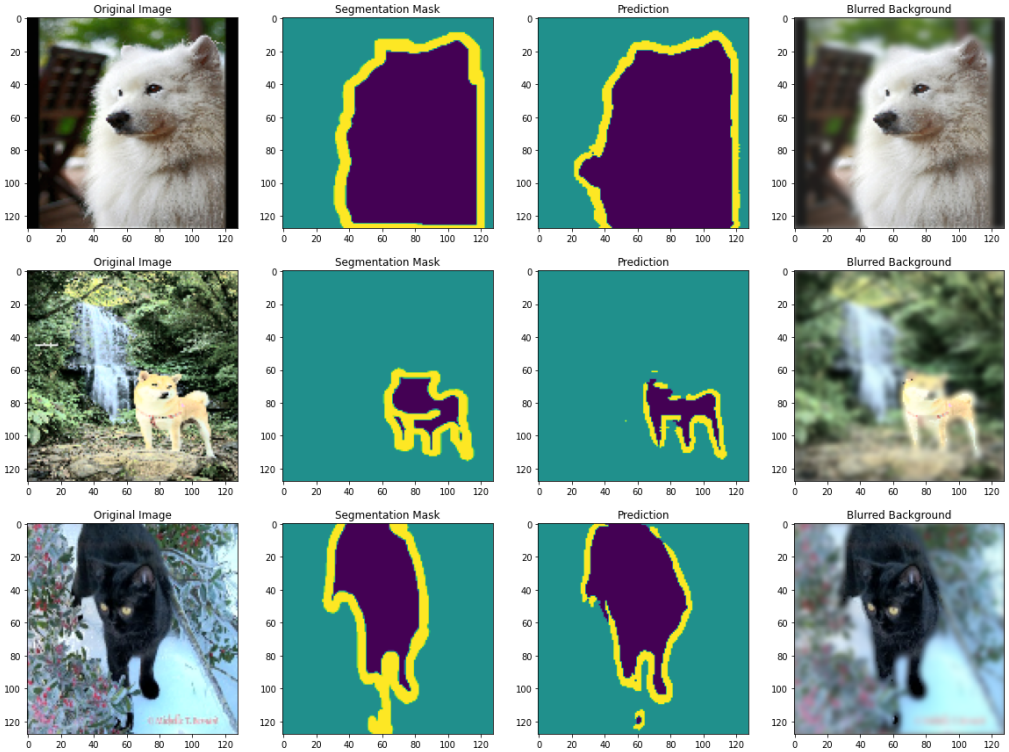


Figure 4: Sample result with blurred background

difficult to intuitively understand what the accuracy of around 88% means in the semantic

segmentation. The best way to grasp the sense is visualizing, which is shown in figure Figure 4. Here, we can see that the prediction is mostly capturing where in the image the animal exists though it seems difficult to distinguish the collar around the neck in the second image, and the foreleg hidden behind the leaves.
Also, blurring the background works very similar to what we see in the Zoom meeting. For this visualization, we set the Gaussian filter size to 11×11. We also implemented a function that simply changes the background with a given image. Figure 5 shows the result.

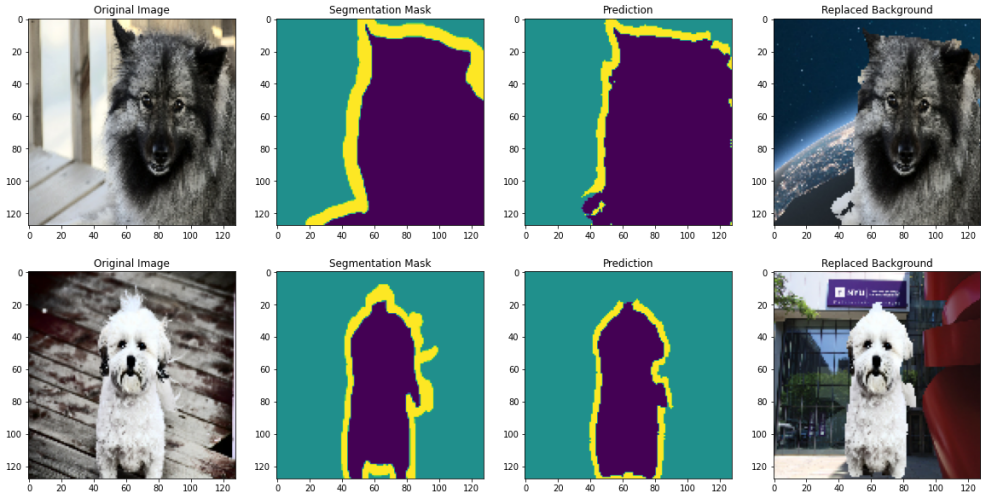


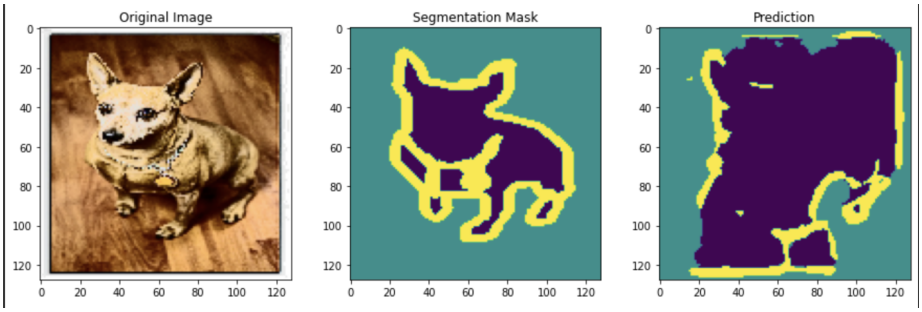
Figure 5: Sample result with replaced background

5 Difficulties and Challenges

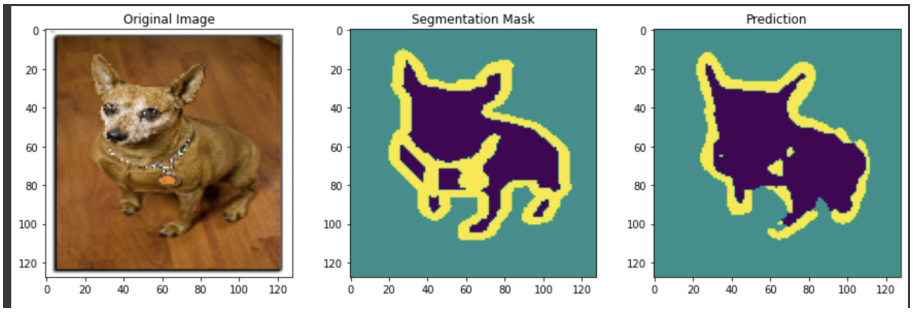
In this section, we will illustrate the difficulties and challenges that we faced and how we conquer the obstacles.

During the dataset choosing phase, we had two candidates. Originally, we planned to use the CoCo dataset. COCO is a large-scale object detection, segmentation and caption dataset. It contains a very comprehensive images with labeled categories. However, CoCo dataset has some drawbacks. First of all, it is too huge. Given that we have limited time and limited resources, we can hardly train the dataset. Secondly, the test dataset of this dataset has no annotations. In addition, the dataset defined 91 classes but only uses 80 of them. It makes the data hard to use and hard to train. Thus, we choose not to use CoCo dataset. In stead, we chose to use the Oxford pet dataset that we introduced in section 3.

During the training phase, we wanted to improve the results of the prediction by refine the contrast of the images. In order to solve it, histogram equalization came to our mind. However, using this method brought us another problem. The algorithm that we used and learned for the equalization is performed on gray scale images which have only 1 color channel. But in this project, we are dealing with RGB images with 3 color channel. We did some research and found that we can actually convert the RGB image into YPbPr scale, doing the equalization and convert it back. It turned out that the training and testing accuracy are not changed after the modification. Thus, we checked the predicted results visually to see if there are any changes.



(a) Image with Histogram Equalization



(b) Image without Histogram Equalization

Figure 6: Histogram Equalization

Unfortunately, As shown in figure 6, histogram equalization cannot give us a better performance visually. In this example figure, although the contrast is enhanced, the brown background actually got mixed up with the color of the dog and lead to a worse result. We discussed this result and came up with one reasoning that, even after the histogram equalization, the order of intensity in each pixel doesn't change. Therefore, the maxpooling layer is very likely to pick the information from the same location no matter histogram equalization is used or not.

Another challenge that we faced is that how to apply filters like Gaussian filters only

to the background of the segmented image. In order to do this, we make use of the prediction mask. Firstly, we use the prediction mask to get the foreground in the original image and store it. Then we apply the filter to the whole image to make the image blurred. After that, we can replace the foreground pixels back by using the stored foreground.

6 Discussion

In this section, we firstly point out strengths and weaknesses of our model. Then, we discuss the performance of our model on randomly picked images from the Internet. We also mention the potential of our model as an edge detection tool.

The strength of our model is, as it is also described in the Approach section, the model considers both the context of the entire image and the local information to make a prediction for each pixel. This is done mainly by maxpooling layers and transposed convolution layer. Note that, unlike the model presented in the paper, we set the padding parameter of convolution layer to "same", and thus, our convolution layers don't downsample the information. The maxpooling layer combined with convolutions enables us to extract the context of the whole image. The transposed convolution layer upsamples the context information, and this makes bypassing local information by concatenate layer possible. One weakness of our model is that the prediction becomes worse when some objects overlaps the animal. As we saw in the Result section, when the collar hides the neck of a dog, or when leaves overlaps the foreleg of a cat, the model often produce wrong predictions.

A fun part of building an image processing application would be applying our model to random images that are not in the original data set. By doing so, we can test the robustness of our model. We tested our model on three images that we picked from the Internet. The results are displayed in the figure 7. First image is a lion. Recall that the training data consists only of dogs and cats. Despite that images of a lion are not used to train the model, it produces a fairly good prediction. Second image is a dog. Again, the model gives a good prediction, as expected. The last example is an image from Tom and Jerry, which is a very different image from our data set because this is a cartoon not a picture. The result is ironical. Even though we trained our image on pictures of dogs and cats, hoping that the model detects a cat correctly, it finds out a mouse and ignores a cat.

Lastly, figure 8 suggests a potential of our model as an edge detector. The left image is made from the prediction of our model, and the right image is created by the Canny edge detector. The advantage of using our model is that it only captures the outline of the lion, and this is the reason we used semantic segmentation for separating an animal from background. We would like to insist that this is the power of pixel-wise classification. On

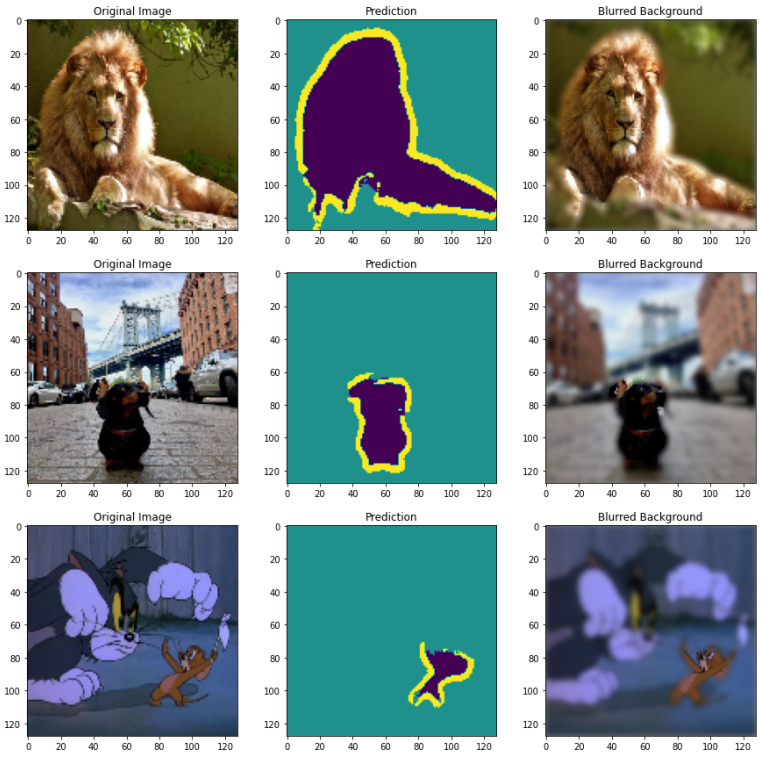


Figure 7: Out of data set images

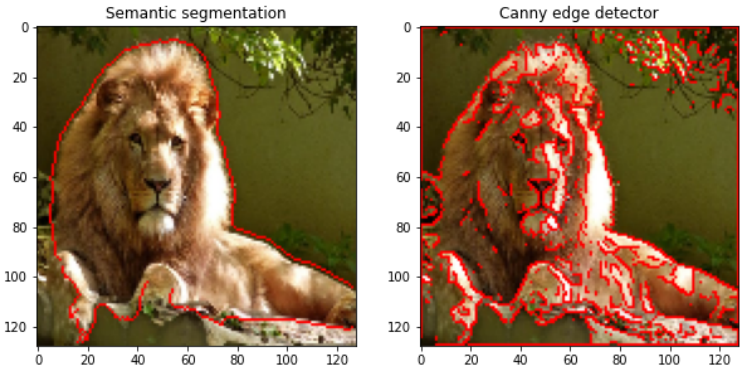


Figure 8: Edge detection

the other hand, Canny detects the edge more sharply and precisely mainly because Canny only focuses on the local information of the image. As a result, Canny detects not only the outline of the lion but also other features of the lion as well as edges in the background. So,

there are pros and cons in both approaches and thus, it is impossible to argue that one approach is superior to the other. It depends on what we want to do with these techniques.

7 Future work

In this section, we will point out several approaches that could further improve our model.

As we mentioned in the discussion section, our model is not good at distinguishing a collar from an animal, for example. We think one of the approaches to overcome this issue would be using images with higher resolution. Usually, an object overlaps the animal is small, and this may be the reason for the model to misclassify it. We developed an intuition that the relative size of the kernel in the convolution layer against the overlapping object may cause this issue. As the kernel size is 3×3 in our model and unfeasible to make it even smaller. Thus, only way to adjust the relative size is making the image larger.

Another interesting approach would be increasing the number of labels. In this project, we used only three labels, but there exists data sets that classify pixels into more than ten labels. One example is called "Cityscapes" data set, which consists of street scenes images with as much as 30 labels. If we use this data, we can blur objects other than cars and pedestrians, for instance.

In the previous section, we discuss the advantages and disadvantages of semantic segmentation and Canny edge detector. Another potent approach is using the stereo with multiple cameras. With stereo, we can compute the distance between objects and cameras. Once we know the distance, the detection task would become easier. Also, the snake can be used for this task. To use the snake, we need to specify the initial location. Though the animals' position in the image vary, the semantic segmentation gives the rough location. So, combining with the semantic segmentation, it may be interesting to use snake for detecting the outline of animals.

We are also curious about how our model works on videos. Considering that the Zoom meeting is a live streaming, semantic segmentation and blurring should be done very quickly. Our model may not fast enough to be used in real-time situations. Although we implemented the Gaussian filter as a combination of vertical and horizontal 1D filters, it takes some time to process even a tiny 128×128 image.

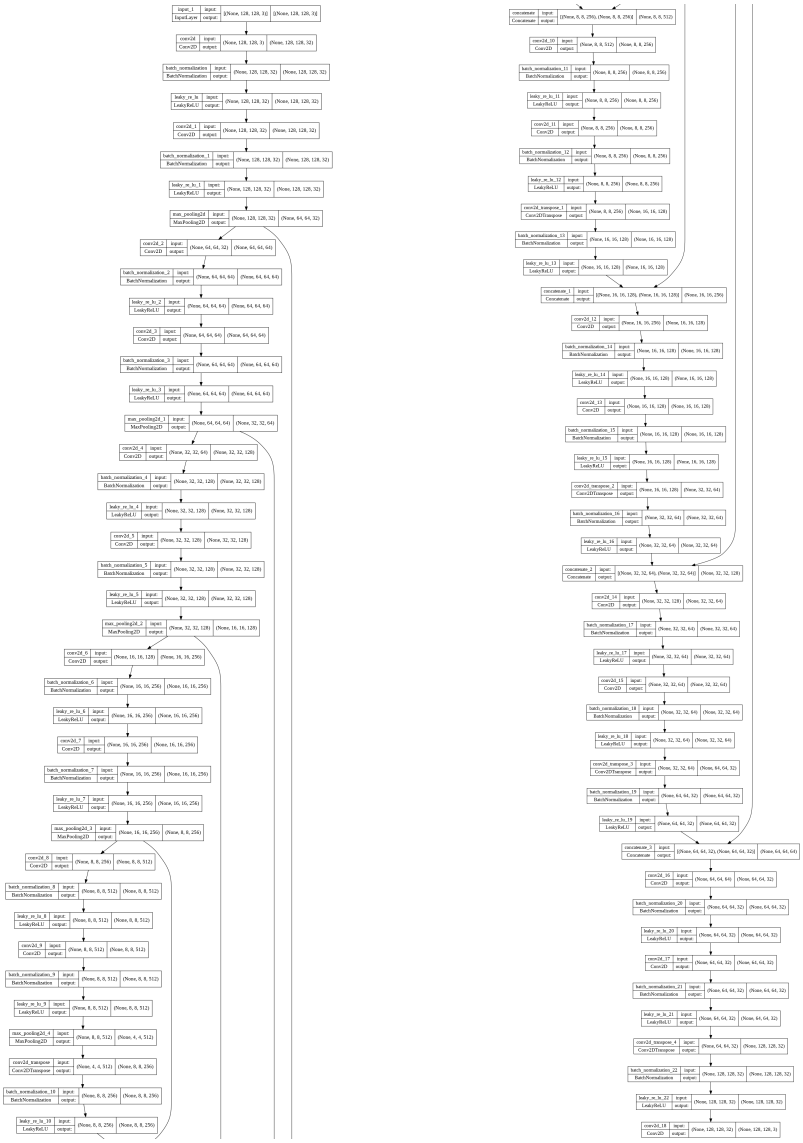
8 Conclusion

So far, we present our own semantic segmentation model and the use of the prediction for manipulating the original image. We successfully show the potential of using semantic

segmentation for separating an animal from background.

9 Appendix

The network architecture.



References

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.